Q1a (Trace-code; parameter-passing): What is the output of the following program? [3 Marks]

```c
int fun(int& a, int b) {
    a += 3;
    b = 3;
    cout << "a=" << a << "b=" << b << "\n";
    return a%b;
}

int main() {
    int a = 7;
    int b = 4;
    int c = fun (a,b);
    cout << "a=" << a << "b=" << b << "c=" << c << "\n";
    return 0;
}
```

Step 1: Contents of the memory locations are:

Step 2: Contents of the memory locations are:

Step 3: The output of the program is:

Q1b (Trace-code; pointers): Show the contents of the memory locations at the specified steps. [3 Marks]

```c
int main () {
    int num[4]; int* p;
    for (int i=0; i < 4; i++) num[i] = 0;
    p = num;   // Show the memory here ... Step 1
    *p = 10;
    p++;  *(p++) = 20;
    *p = 30;    // Show the memory here ... Step 2
    p = &num[3]; *p = 40;
    *(p-1) = 30;
    for (int n=0; n<4; n++)
        cout << num[n] << ", ";
    return 0;
} // Show the final output ... Step 3
```

Step 1: Contents of the memory locations are:

Step 2: Contents of the memory locations are:

Step 3: The output of the program is:
Q2 (Write code; Strings): Write a C++ function that takes an array of strings as input, and returns the smallest string in dictionary order. For example, if the array has strings {abcd, abc, bc, acd}, your function should return 'abc'. You should use C++ string type and its functions. If you are not familiar with C++ string, then you may use the C strings library functions, or any other way, with a penalty of two marks. State any additional assumptions that you need to make. [8 Marks]
Q3 (Write code; Linked Lists): Write a function, `sortedIntersect()`, that takes two lists sorted in increasing order, and returns a new linked list having the intersection of the two lists. For example, if the original lists are given as {1,2,3,4} and {2,3, 23}, then `sortedIntersect()` should return a list containing {2, 3}.

You may use assume the following as given:

```c
struct node { int num; node * next; }; // each node of the list has one number and a pointer.
node* append(node *list, int item); // function that appends item at the end of a given list and returns
        // a pointer to its head.
int main(); // Main program that creates two lists L1 and L2 and calls sortedIntersect() appropriately.
```

You need to do:

1. Write the pseudo-code (logic) for `sortedIntersect()`.
2. Write the C++ code for above: `node* sortedIntersect(node* L1, node* L2);`

[4 Marks] [6 Marks]
Q4 (Debug-code; matrices): Your friend wrote the program below to find all the zeros in an 8*8 matrix and replace the corresponding rows and columns with zeros. A sample input and output are shown below. Fill in the two places of the missing code as indicated. [6 Marks]

<table>
<thead>
<tr>
<th>Example Input Matrix</th>
<th>Example Output Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3 4 5 6 7 8</td>
<td>1 0 3 4 0 6 0 8</td>
</tr>
<tr>
<td>2 0 2 3 4 5 6 9</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>3 1 5 7 8 4 2 3</td>
<td>3 0 5 7 0 4 0 3</td>
</tr>
<tr>
<td>4 3 8 7 0 3 3 5</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>5 2 3 4 5 6 1 7</td>
<td>5 0 3 4 0 6 0 7</td>
</tr>
<tr>
<td>6 7 6 6 9 1 8 9</td>
<td>6 0 6 6 0 1 0 9</td>
</tr>
<tr>
<td>7 0 7 4 3 2 0 2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>8 9 5 2 3 4 5 6</td>
<td>8 0 5 2 0 4 0 6</td>
</tr>
</tbody>
</table>

```cpp
#include <iostream>
using namespace std;

const int n = 8;
struct locations { //keep track of rows and columns that need to be nullified
    int row[n], col[n];
};

void nullifyRow(int A[n][n], int row) { for (int i=0; i<n; i++) A[row][i] = 0; }
void nullifyCol(int A[n][n], int col) { for (int i=0; i<n; i++) A[i][col] = 0; }

locations findZeros(int A[n][n]); //Find the zeros in the given matrix and record their location in the struct.
// It first initializes locations to all 1s and then changes a particular entry to 0 as appropriate.
// This is the function declaration. Fill in this function definition on the next page.

int main() {
    locations h; int Matrix[n][n]; int i, j;
    for (i=0; i<n; i++)
        for (j=0; j<n; j++) cin >> Matrix[i][j];

    h = findZeros(Matrix);

    // Call nullifyRow, nullifyCol as per the entries in h.
    // Fill in this part of the code

    for (i=0; i<n; i++)
        for (j=0; j<n; j++) cout << Matrix[i][j] << " ";
    cout << endl;
}
```
Space for continuation of answer to Q4:

locations findZeros(int A[n][n]) { // Fill in this code